Arrays
Friday Four Square!
Today at 4:15PM, outside Gates.
A Different Way to Store Data

- Last time, we saw the `ArrayList` as a way to store lots of data.
  - Lines of text.
  - US cities!
- Java also supports a concept called the `array` that can used to store lots of data.
Recapping `ArrayList`

- An `ArrayList` stores a **sequence** of multiple objects.
  - Can access objects by index by calling `get`.
- All stored objects have the same type.
  - You get to choose the type!
- Must store objects; primitive types not allowed.
- Can grow as long as it needs.
Introducing Arrays

An array stores a sequence of multiple objects.
- Can access objects by index using square brackets (more on that soon).
- All stored objects have the same type.
  - You get to choose the type!
- Can store any type, even primitive types.
- Size is fixed; cannot grow once created.
Basic Array Operations

• To create a new array, specify the type of the array and the size in the call to `new`:

  `Type[] arr = new Type[size]`

• To access an element of the array, use the square brackets to choose the index:

  `arr[index]`

• To read the length of an array, you can read the `length` field:

  `arr.length`
Default Values in Arrays

• When creating an array:
  • `int`, `double`, `char`, etc. default to 0,
  • `boolean` defaults to `false`, and
  • Objects default to `null`. 
A Nuance of Pass-by-Reference

- Arrays are objects, so they are passed by reference.
- The **elements** of an array, like the fields of an object, can be modified inside of a method.

```java
int[] arr = new int[5];
fillArray(arr);
```
A Nuance of Pass-by-Reference

- Arrays are objects, so they are passed by reference.
- The **elements** of an array, like the fields of an object, can be modified inside of a method.

```java
private void fillArray(int[] arr) {
    for (int i = 0; i < arr.length; i++) {
        arr[i] = i;
    }
}
```
A Nuance of Pass-by-Reference

- Arrays are objects, so they are passed by reference.

- The **elements** of an array, like the fields of an object, can be modified inside of a method.

```java
private void fillArray(int[] arr) {
    for (int i = 0; i < arr.length; i++) {
        arr[i] = i;
    }
}
```
A Nuance of Pass-by-Reference

- Arrays are objects, so they are passed by reference.
- The **elements** of an array, like the fields of an object, can be modified inside of a method.

```java
int[] arr = new int[5];
fillArray(arr);
```

![Diagram showing an array and its elements]
Why Arrays?

- Arrays are excellent for representing a fixed-size list of **buckets**.
- We can store values in the appropriate bucket by looking up the bucket by index.
How many people need to be in a room before two of them will share a birthday?
The Birthday Paradox

• In a room of 23 people, there is a 50% chance that two of them have the same birthday.

• More generally, if you have an $n$-sided die, you only need to roll it around $\sqrt{2n}$ times before you have a 50% chance of getting the same outcome twice.

How many people do you need, on average, for three people to share a birthday?
Sound Processing
The Physics of Sound

- Sound is a wave that propagates through a medium.
- The frequency of the wave is how closely packed together the peaks are.
  - Corresponds to pitch.
- The amplitude of the wave is how tall the peaks are.
  - Corresponds to loudness.
Representing Sound

- The computer can represent a sound by storing the sound wave.
Representing Sound

- The computer can represent a sound by storing the sound wave.
Representing Sound

- The computer can represent a sound by storing the sound wave.

- Unfortunately, the wave is continuous, so the computer cannot store it completely.
Representing Sound

- The computer can represent a sound by storing the sound wave.

\[ \text{Wave graph} \]

- Unfortunately, the wave is continuous, so the computer cannot store it completely.

**Idea:** Sample points from the sound wave and store those instead.
Representing Sound

- The computer can represent a sound by storing the sound wave.

- Unfortunately, the wave is continuous, so the computer cannot store it completely.

- **Idea:** Sample points from the sound wave and store those instead.
Representing Sound

- The computer can represent a sound by storing the sound wave.

- Unfortunately, the wave is continuous, so the computer cannot store it completely.

- **Idea:** Sample points from the sound wave and store those instead.
Representing Sound

• The computer can represent a sound by storing the sound wave.

• Unfortunately, the wave is continuous, so the computer cannot store it completely.

• **Idea**: Sample points from the sound wave and store those instead.
Representing Sound

- The computer can represent a sound by storing the sound wave.

- Unfortunately, the wave is continuous, so the computer cannot store it completely.

- **Idea:** Sample points from the sound wave and store those instead.
The Sampling Rate

- The **sampling rate** for a sound clip is the frequency at which the wave's intensity is recorded.
  - Measured in hertz (Hz).
- Example: If sampling rate is 44,100Hz, there are 44,100 samples per second.
- High sampling rate makes for better sound.
- Low sampling rate uses less storage space.
Generating Sound

- Today, we'll use Princeton's StdAudio class to play sounds.
- Each sound clip is represented as a double[], where each entry is between -1 and +1.
- We can play the sound by calling 
  ```
  StdAudio.play(soundClip)
  ```
Creating a Sine Wave

- To make a sine wave with frequency $f$, we want to sample from the wave:
  $$\sin(2\pi x f)$$

- However, since time is scaled by the sampling rate, the wave we want is:
  $$\sin\left(\frac{2\pi x f}{\text{SAMPLING\_RATE}}\right)$$
Equal-Loudness Contours

Equal-loudness contours (red) (from ISO 226:2003 revision)
Original ISO standard shown (blue) for 40-phon

Loading Sound

• The `StdAudio` class also has a function for loading sound from a `.wav` file:
  
  ```java
  double[] sound = StdAudio.read(filename)
  ```

• Requires the sound file to use 44.1KHz sampling rate.